REMARKS

This Amendment is in response to the Office Action of February 15, 2005 in which the Examiner rejected claims 31, 32, 34-38 as allegedly anticipated by Endicott U.S. Patent No. 5,404,525.

The Examiner rejected claim 33 as unpatentable over Endicott '525.

The Examiner's rejection is respectively traversed for the reasons set forth below.

Endicott describes a method for routing on an object of an enhanced Object Oriented Programming (OOP) environment, such that it becomes possible to create versions of object classes, and to change the version of an object instance from one version to another, without having to recompile everything as in C++, and without performance penalties as in Smalltalk. To achieve this objective, Endicott uses various data structures as a way to locate (i.e., find the address of) objects, interfaces, and methods.

The present invention deals with the problem of creating and instantiation of composite object types in a system for automatic control of real world entities, for example in industrial processes (the present invention uses the term "object type" whereas Endicott uses the term "object class". These terms are more or less equivalent). A fundamental part of the method is the concept of object structures. Unlike traditional object class hierarchies, these structures organize object instances (not classes), and can be based on any kind of relationship between object instances. Object structures are used by process engineers and operators, as well as by application programs, as a way to navigate among the typically very large amount of objects found in a process control system.

A composite object type in the present invention is an object type (class), which holds references to other object types. Being references to object types but not actual instances, these references are called formal instances. The formal instances are arranged in groups, each group referring to a structure. When the composite object type is instantiated, for each group and formal instance an actual instance is created and placed in the structure referred to from the group. Thus, when instantiated, a composite object type results not only in a single object instance, but in multiple object instances of different object types which are placed in different structures. In addition, as these object instances are placed in structures, they may be modified to adapt to how they relate to other object instances in the structures where they are placed. This is particularly useful in a control system, where the exact behavior of a control function depends on in which context it is being used.

These objectives as well as the solutions described in the present invention are completely different from those described by Endicott.

The Examiner made specific rejections to the claims. What follows is a detailed side by side comparison of the Examiner's comments, where the numbers in the left column represent the paragraph numbers in the office action, with the Applicant's responses.

Claim rejections		**Inventor's comments**
1. Claims 31, 36, and 37	Endicott discloses a method of claim 31, a system of claim 36, and a computer program comprising program code means of claim 37 for performing steps of a method for automatic control of real world entities, wherein the real world entities are represented as instances of objects, and wherein the control of an	Endicott does <u>not</u> disclose a method, system, or program code for automatic control of real world entities. In fact, Endicott's only mention of the word "control" is in column 1 line 45, in relation to lack of control over development and maintenance of large and complex computer programs. Endicott's invention deals with the problem of changing versions of objects without having to recompile all object instances

Serial No.: 09/973,069
Docket No.: 66848-001

Claim rejections		Inventor's comments
	individual entity depends on the relationship of the entity to other entities (column 5, line 15 - column 6, line 48).	(which is needed with e.g. C++), and without the performance penalties seen e.g. with Smalltalk In column 5, line 15 – column 6, line 48, Endicott describes the internal organization of a computer, and the concept of class hierarchies in object oriented programming (OOP). These are well known concepts in the prior art. There is no reference in Endicott to a method for control of real world entities where the control of an individual entity depends on how it relates to other entities as in the present invention.
	a. Creating at least two structures, wherein each structure is based on a certain type of relationship between object instances (column 5, lines 42-68).	Column 5, lines 42 -47 read: "For example, portions of client programs 120 and operating system 135 will typically be loaded into primary memory to execute, while source data files will typically be stored on magnetic or optical disk storage devices." In column 5 lines 48-68 Endicott describes the concept of object class hierarchies. An object class hierarchy is a structure of object classes based on class relationships, i.e., it is built up from super classes and subclasses. In object oriented technology it is very important to distinguish between object classes and object instances. A class defines the characteristics (primarily data and methods) of all instances of that class, while instances are the actual objects. In Endicott Fig. 2, John and Joe are instances of the object class Engineer. An object class can be a specialization (subclass) of another object class (super class). In Endicott Fig. 2, the class Personnel has three subclasses: Lawyer; Engineer; and Manager. Step a of claim 1 of the present invention describes the step of creating at least to structures based on certain types of relationships between object instances, not object classes. Examples of such relationships are given in the text and in subsequent claims. Endicott does not involve any such structures, and the reference to Endicott column 5 lines 42-68

Serial No.: 09/973,069
Docket No.: 66848-001

Claim rejections		Inventor's comments
		is thus believed to be irrelevant.
	b. Creating at least one composite object type comprising at least two formal instances, wherein each formal instance contains information about instantiation of a certain object type, and wherein at least one such object type defines a function for control of a real world entity (column 5, line 58 column 6, line 27 and column 12, line 57 - column 13, line 35).	In column 5 line 58 – column 6 line 27 Endicott describes a class hierarchy (see above). Step b of claim 1 of the present invention describes the step of creating a composite object, which is an object type that "includes" at least two other object types. This "inclusion" is not a class relationship (a building can contain a pump, but Pump is obviously not a specialization Building). The structures described in the present invention are thus not class hierarchies. Step b further includes the element that at least one of the thus included object types defines functions for control of a real world entity. Endicott does not mention control of real world entities. In column 12, line 57 – column 13, line 35, Endicott describes how a method program is located when called by a client program. Endicott describes how the object class is located, the object instance loaded, the proper interface is located and the update method invoked. Endicott uses various data structures as a way to locate (i.e. find the address of) objects, interfaces, and methods. These data structures are described in overview in column 3 line 44 ff, and in detail in the figures 2, 4A-D, 5, 9A-1, 9A-2, and 10A. These data structures are clearly not examples of structures based on relationships between object instances, such as functional containment or location.
	c. Locating each formal instance in at least one of two groups of formal instances, wherein each group is associated with a structure, and at least two groups are associated with different structures (column 5, line 55 - column 6, line 48).	In column 5, line 55 - column 6, line 48, Endicott describes the concept of object class hierarchies. The structures referred to in step c of claim 1 of the present invention are structures between object instances (see above)
	d. Instantiation of the composite object type, wherein for each group of formal instances corresponding real	In column 13, lines 47-68, Endicott describes an example where a new version of an object class is created. When the new version is created, new versions of all

Claim rejections		Inventor's comments
	world object instances are created and located in the structure with which the group is associated (column 13, lines 47-68)	subclasses are also created. Endicott further allows instances of both versions of the object class to exist simultaneously (engineer Sam is promoted to the new version, while engineers John and Joe remain with the previous version). The objective of the present invention is completely different, namely to make it possible to define composite object types (classes) which, when instantiated, result not only in a single object instance, but in multiple object instances of different object types, where the instances are placed in different structures as defined in the definition of the composite object type. Step d of claim 1 describes how this instantiation and placement is done. This is clearly different from what Endicott describes in column 13, lines 47-68.
	e. Changing at least one object instance thus created, said object instance defining a function for control of a real world entity, such that said function for control is adapted to the relationship of the object instance to other object instances in at least one of the resultant structures such that the control of the corresponding real world entity is adapted to the relationship of said entity to other entity (column 14, lines 39-60).	In column 14 lines 39-60, Endicott describes an example of how an object is promoted from one version of its object class to another version. Endicott's objective is that such changes should not affect client or method programs that deal with parts of the object that are not affected by the change (column 14 line 30-39). The objective of the present invention is entirely different, namely to make it possible to define object types (classes) which, when instantiated, result not only in a single object instance, but in multiple object instances of different object types, where the instances are placed in those structures (of object instances) that are pointed out in the definition in the object type. Step e of claim 1 describes how a control function defined by such object instance is adapted to fit with the relationships to other instances in structures where it is placed. An example of where such adaptation is needed is the control function for a valve, which needs to behave differently when it is part of a level control loop from when it is part of a flow control loop. There is no mention of control functions for control of a real world entity in Endicott,

Claim rejections		Inventor's comments
		and specifically, there is no mention of how such control function is adapted to how the entity is related to other entities when the object is placed in object structures. In particular, this is not described or disclosed in Endicott column 14, lines 39-60.
2. Claim 32	Endicott discloses all the limitations of claim 31. In addition, Endicott discloses a method wherein at least one formal instance is of a composite object type (column 13, lines 47-68 and Fig 2).	As argued above, Endicott does not disclose any of the limitations of claims 31, 36, and 37. A composite object type in the present invention is an object type (class), which holds references to other object types. Being references to object types but not actual instances, these references are called formal instances. Endicott does not disclose or describe any concept that is similar to the concept of composite object types with formal instances according to the present invention. The cited lines from Endicott are part of the section "ADDITION OF A NEW INSTANCE VARIABLE DEFINITION TO A NOM CLASS". An "instance variable" is part of the anatomy of an individual object – it holds data that is specific to an instance, such as the name of an employee. In Endicott's example, a new instance variable is added to the class Personnel, thus creating a new version of that class, Personnel_II. The new instance variable is intended to hold the number of patents each employee (instance of Personnel_II) has produced. This is an example of changing an object class and is not in any way related to the concept of composite object types and formal instances. Endicott uses the term "composite", but only in relation to the data structures that make up the implementation of the NOM object oriented environment. In column 3 lines 44 ff, Endicott describes three key composite data structures of the NOM object oriented environment: the object structure, the interface table, and the method table. These structures are related to the implementation of Endicott's object oriented environment. In particular, Endicott's object structure is a table that

Serial No.: 09/973,069
Docket No.: 66848-001

Claim rejections		Inventor's comments
		contains data that characterizes an object, and location information about (pointers to) the interface table. Note that "location" in Endicott refers to a location (address, pointer, or index) in the computer system, whereas "location" in the present invention refers to the physical location of real world entities, such as a pump is placed in a room that is part of a building. Endicott Fig. 2 shows a classical object class hierarchy. There is no example of a composite object type with formal instances as defined in the present invention,.
3. Claim 34	Endicott discloses all the limitations of claim 31. In addition, Endicott discloses a method wherein a formal instance comprises a description of how the properties of the corresponding object type are to be changed when a real world instance is created (column 12, line 57 - column 13, line 22 and column 15, line 54 - column 16, line 13).	This claim should read "... to be changed when an actual instance is created". The invention does not create real world instances. As noted above, Endicott does not disclose any of the limitations of claims 31, 36, and 37. The concept of formal instances is related to the concept of composite objects. As argued above, Endicott does not disclose or describe any concept that is equal or similar to the concept of composite object types according to the present invention. Specifically, Endicott does not describe or disclose how object types are changed when actual instances are created from formal instances. In column 12 line 57 - column 13 line 22, Endicott describes how a method program is located when called by a client program. The example assumes that the client program wants to update the instance variable Salary for the object instance John. Endicott describes how the object class is located, the object instance loaded, the proper interface is located and the update method invoked. Invoking a method of an object (including localizing the class, loading the object, finding the interface and invoking a method) is fundamentally different from creating an object instance. The cited lines are therefore irrelevant to claim 34 of the present invention. Column 15, line 54 – column 16, line 13 is

Claim rejections		Inventor's comments
		part of the section "NOM METAMORPHOSIS". Endicott defines this as changing the version of a particular object instance (column 15 lines 44-53). This is clearly different from creating an instance of a composite object type wherein a formal instance comprises a description of how its corresponding object type shall be changed when the actual instance is created.
4. Claim 35	Endicott discloses all the limitations of claim 31. In addition, Endicott discloses a method wherein at least one formal instance represents a group of formal instances (column 5, line 55 - column 6, Line 10).	As argued above, Endicott does not disclose any of the limitations of claims 31, 36, and 37. Endicott does not disclose or describe any concept that is similar to the concept of composite object types according to the present invention. Specifically, Endicott does not disclose or describe how a composite object is defined as comprising formal instances. Hence, Endicott does not describe how a formal instance can represent a group of formal instances. In the present invention, the step of letting a formal instance represent a group of formal instances provides a way of referring to the entire group by referring to the one formal instance that represents it. In column 5, line 55 - column 6 line 10, Endicott describes an object class hierarchy and class inheritance. It is not understood how this can be interpreted as being related to having a formal instance representing a group of formal instances.
5. Claim 38	Endicott discloses all the limitations of claim 36. In addition, Endicott discloses a computer program wherein the program code means are stored in a computer readable medium (column 5, lines 1-10).	Endicott does not disclose a computer program as defined in claim 38 of the present invention. Endicott column 5 lines 1-10 are part of the section "BRIEF DESCRIPTION OF THE DRAWINGS", and specifically describe drawings 7, 8A, 8B, 9A, 9B, 10A, and10B. Neither of these figures describes a computer program wherein the program code means are stored in a computer readable medium.
6. Claim 33	Claim 33 is rejected under 35 U.S.C. 103(a) as being unpatentable over Endicott et al (U.S. 5,404,525).	As pointed out above, the present invention deals with structures of object instances. These are fundamentally different from class hierarchies, in that

Serial No.: 09/973,069
Docket No.: 66848-001

Claim rejections		Inventor's comments
	Endicott discloses all the limitations of claim 31. In addition, Endicott discloses a method wherein one structure is based on functional properties of the real world entities and another structure is based on the physical location of the real world entities (column 2, lines 7-35 and column 6, lines 12-48). Note that Endicott does not explicitly disclose that his invention has a structure that is based on the physical location of real world entities. However, Endicott implies in column 2, lines 7-35 that it was known in the art at the time the applicant invention was made. Endicott uses the example of a Dog that had eyes and four legs. In this matter, Endicott implies that the objects "eyes" and "legs" are located with the object "Dog." One of ordinary skill would likely incorporate this background teaching with Endicott's as it would allow for better encapsulation and reusability of programmed entities (column 1, line 5g - column 2, line 6).	they are based on relationships between object instances, not between object classes. Examples of relationships are functional containment and location. Examples of functional containment are a processing line, which includes a tank, which includes a level control loop, which includes a pump and a valve. It is obvious that the pump and the valve are not specializations of a control loop, which in turn is not a specialization of a tank, which is not a specialization of a processing line. Endicott does not describe the parts of a Dog as objects. In fact, Endicott does not mention "eyes" and "legs" as being parts of the object Dog at all. Endicott describes the object class Canine, which defines data such as name, color, number of eyes, number of legs, etc. (column 2 lines 7-35). This data is part of the object Canine, not separate objects that are associated with it. Endicott does not mention anything about organizing object instances into structures. Further, as shown above, the statement that "eyes" and "legs" are objects that are located with the object "Dog" is thus believed to be incorrect.

Serial No.: 09/973,069
Docket No.: 66848-001

It is believed that the invention is patentable over the art record, it is respectfully requested that the Examiner reconsider his rejection of the claims, the allowance of which is earnestly solicited.

Respectfully submitted,

DYKEMA GOSSETT PLLC

By:

John P. DeLuca
Registration No. 25,505
Franklin Square, Third Floor West
1300 I Street, N.W.
Washington, DC 20005-3353
(202) 906-8626

DC01\92337.1
ID\JPD